

EAST Search History

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L1	725	(determin\$3 or detect\$3 or check\$3 or find\$3) same (query or RFQ or request or RFP) same (comparison or BOT)	EPO; JPO; DERWENT	OR	ON	2006/05/02 18:25
L2 <i>Full</i>	8	(determin\$3 or detect\$3 or check\$3 or find\$3) same (query or RFQ or request or RFP) same (comparison or BOT) same price	EPO; JPO; DERWENT	OR	ON	2006/05/02 18:27
L3 <i>Full</i>	43	comparison near4 shopp\$3	EPO; JPO; DERWENT	OR	ON	2006/05/02 18:27

EAST Search History

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L4	4565	705/26.ccls.	US-PGPUB; USPAT	OR	ON	2006/05/02 13:56
L5	1493	705/27.ccls.	US-PGPUB; USPAT	OR	ON	2006/05/02 13:56
L6	227	705/20.ccls.	US-PGPUB; USPAT	OR	ON	2006/05/02 13:56
L7	2366	705/37.ccls.	US-PGPUB; USPAT	OR	ON	2006/05/02 13:57
L8	734	705/400.ccls.	US-PGPUB; USPAT	OR	ON	2006/05/02 13:57
L9	3588	705/14.ccls.	US-PGPUB; USPAT	OR	ON	2006/05/02 13:57
L10	<i>Full</i> 27	(compar\$4 near (site or URL or address)) same (compar\$5 near price)	US-PGPUB; USPAT	OR	ON	2006/05/02 13:57
L11	<i>Full</i> 7	4 and 10	US-PGPUB; USPAT	OR	ON	2006/05/02 13:58
L12	<i>Full</i> 1	5 and 10	US-PGPUB; USPAT	OR	ON	2006/05/02 14:18
L13	0	6 and 10	US-PGPUB; USPAT	OR	ON	2006/05/02 13:58
L14	<i>Full</i> 3	7 and 10	US-PGPUB; USPAT	OR	ON	2006/05/02 14:18
L15	0	8 and 10	US-PGPUB; USPAT	OR	ON	2006/05/02 13:58
L16	<i>Full</i> 4	9 and 10	US-PGPUB; USPAT	OR	ON	2006/05/02 14:22

EAST Search History

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
S1	1	"20020016779".pn.	US-PGPUB; USPAT	OR	ON	2005/05/23 12:41
S2	14	(compar\$4 near (site or URL or address)) same (compar\$5 near price)	US-PGPUB; USPAT	OR	ON	2005/05/20 16:16
S3	16	(query or request or message) near15 (receiv\$3 or transmit\$4 or sent or send of forward\$3) near (compar\$4 near (site or URL or address))	US-PGPUB; USPAT	OR	ON	2005/05/20 16:20
S4	0	S3 same (price or cost)	US-PGPUB; USPAT	OR	ON	2005/05/20 16:19
S5	3	(query or request or message) near15 (receiv\$3 or transmit\$4 or sent or send of forward\$3) near (compar\$4 near (site or URL or address)) and price	US-PGPUB; USPAT	OR	ON	2005/05/20 16:19
S6	1	(query or request or message) near15 (receiv\$3 or transmit\$4 or sent or send of forward\$3) same ((compar\$4 near (site or URL or address)) near14 price)	US-PGPUB; USPAT	OR	ON	2005/05/20 16:22
S7	4	(query or request or message) near15 (receiv\$3 or transmit\$4 or sent or send of forward\$3) same ((compar\$4 near6 (site or URL or address)) near14 price)	US-PGPUB; USPAT	OR	ON	2005/05/20 16:22
S8	52	("6012090").URPN.	USPAT	OR	ON	2005/05/20 16:27
S9	2107	ng.in.	US-PGPUB; USPAT	OR	ON	2005/05/20 16:28
S10	31670	"705"/\$.ccls.	US-PGPUB; USPAT	OR	ON	2005/05/20 16:28
S11	14	S9 and S10 and price	US-PGPUB; USPAT	OR	ON	2005/05/20 16:28
S12	13	("6405175").URPN.	USPAT	OR	ON	2005/05/20 16:30
S13	0	((reverse or dutch) near4 auction) and (((display\$3 or show\$3) near5 competi\$4) near4 price)	US-PGPUB; USPAT	OR	ON	2005/05/23 12:42
S14	1	((reverse or dutch) near4 auction) and (((display\$3 or show\$3) near5 competi\$4) near4 price)	US-PGPUB; USPAT	OR	ON	2005/05/23 12:43
S15	3	((reverse or dutch) near4 auction) and (((display\$3 or show\$3) near10 competi\$4) near4 price)	US-PGPUB; USPAT	OR	ON	2005/05/23 12:45

EAST Search History

S16	749	((reverse or dutch) near4 auction)	US-PGPUB; USPAT	OR	ON	2005/05/23 12:45
S17	91	(((display\$3 or show\$3) near10 competi\$4) near4 price)	US-PGPUB; USPAT	OR	ON	2005/05/23 12:45
S18	3	S16 and S17	US-PGPUB; USPAT	OR	ON	2005/05/23 12:45
S19	742	((reverse or dutch) near2 auction)	US-PGPUB; USPAT	OR	ON	2005/05/23 12:45
S20	51	(((display\$3 or show\$3) near10 competi\$4) near4 (bid or price))	US-PGPUB; USPAT	OR	ON	2005/05/23 12:46
S21	4	S19 and S20	US-PGPUB; USPAT	OR	ON	2005/05/23 12:46
S22	1	("6647373").URPN.	USPAT	OR	ON	2005/05/23 12:59
S23	34	("4799156" "4992940" "5664115" "5758328" "5794207" "5794219" "5826244" "5835896" "5842178" "5845265" "5890138" "5897620" "5905975" "5950178" "5966699" "5995947" "6012045" "6014644" "6021398" "6023685" "6023686" "6026383" "6044363" "6055518" "6058379" "6058417" "6216114" "6230146" "6230147" "6260024" "6301574" "6366891" "6397197" "6415269").PN.	US-PGPUB; USPAT; USOCR	OR	ON	2005/05/23 13:14
S24	3	(list near5 (comparison near4 site)) same price same (maintain\$3 or stor\$3)	US-PGPUB; USPAT; USOCR	OR	ON	2005/05/23 15:45
S25	1	"6457030".pn.	US-PGPUB; USPAT; USOCR	OR	ON	2005/05/23 16:34
S26	58	split\$4 near10 ((web near1 site) or website)	US-PGPUB; USPAT; USOCR	OR	ON	2005/05/23 16:34
S27	1403	(compar\$4 near15 shop\$4)	US-PGPUB; USPAT	OR	ON	2005/11/08 17:13
S28	12848	(modif\$5 or discount\$3 or reduc\$3 or chang\$3) near3 pric\$3	US-PGPUB; USPAT	OR	ON	2005/11/08 17:14
S29	277	S27 and S28	US-PGPUB; USPAT	OR	ON	2005/11/08 17:15
S30	5358	best near5 (price or offer)	US-PGPUB; USPAT	OR	ON	2005/11/08 17:15

EAST Search History

S31	Full 60	S29 and S30	US-PGPUB; USPAT	OR	ON	2005/11/08 17:15
S32	9	feedback same (best near10 price) same (bidder or supplier or store or seller)	US-PGPUB; USPAT	OR	ON	2005/11/08 19:15
S33	Full 9	feedback same (best near19 price) same (bidder or supplier or store or seller)	US-PGPUB; USPAT	OR	ON	2005/11/08 18:29
S34	15	feedback same ((best or lowest) near19 price) same (bidder or supplier or store or seller)	US-PGPUB; USPAT	OR	ON	2005/11/08 19:14
S35	Full 6	S34 not S33	US-PGPUB; USPAT	OR	ON	2005/11/08 18:29
S36	15	feedback same ((best or lowest) near39 price) same (bidder or supplier or store or seller)	US-PGPUB; USPAT	OR	ON	2005/11/08 19:14
S37	9	feedback same (best near20 price) same (bidder or supplier or store or seller)	US-PGPUB; USPAT	OR	ON	2005/11/08 19:15
S38	Full 9	feedback same (best near40 price) same (bidder or supplier or store or seller)	US-PGPUB; USPAT	OR	ON	2005/11/08 19:15
S39	7	feedback same (lowest near40 price) same (bidder or supplier or store or seller)	US-PGPUB; USPAT	OR	ON	2005/11/08 19:15
S40	Full 16	("6405175").URPN.	USPAT	OR	ON	2005/11/08 19:17
S41	6	("6076070").URPN.	USPAT	OR	ON	2005/11/08 19:21
S42	0	priegrabber	US-PGPUB; USPAT	OR	ON	2005/11/08 19:21
S43	Full 3	pricegrabber	US-PGPUB; USPAT	OR	ON	2005/11/08 19:21
S44	Full 39	comparison near3 shopping near3 (site or page)	US-PGPUB; USPAT	OR	ON	2005/11/08 19:22
S45	12	S44 and feedback	US-PGPUB; USPAT	OR	ON	2005/11/08 19:22
S46	9	("4002886" "4888709" "5151684" "5172314" "5797131" "5870714" "5873069" "5963133" "5988498").PN. OR ("6703934"). URPN.	US-PGPUB; USPAT; USOCR	OR	ON	2005/11/08 19:24

EAST Search History

S47	3	(bidder or supplier or retailer or manufacturer or seller) near5 (request\$3 or seek\$3 or learn\$3 or determin\$3) near25 ((feedback or comparative or competitor\$2 or competing) near2 price)	US-PGPUB; USPAT	OR	ON	2005/11/09 15:24
S48	5	(bidder or supplier or retailer or manufacturer or seller) near15 (request\$3 or seek\$3 or learn\$3 or determin\$3) near25 ((feedback or comparative or competitor\$2 or competing) near2 price)	US-PGPUB; USPAT	OR	ON	2005/11/09 15:24
S49	2	S48 not S47	US-PGPUB; USPAT	OR	ON	2005/11/09 18:22
S50	742220	(list near25 bots) near "30" (supplier or seller or store)	US-PGPUB; USPAT	OR	ON	2005/11/09 18:23
S51	742220	(list near5 bots) near "30" (supplier or seller or store)	US-PGPUB; USPAT	OR	ON	2005/11/09 18:23
S52	7948	maintain near5 ((list near5 bots) near "30" (supplier or seller or store))	US-PGPUB; USPAT	OR	ON	2005/11/09 18:24
S53	7948	maintain near5 ((list near5 bots) near "10" (supplier or seller or store))	US-PGPUB; USPAT	OR	ON	2005/11/09 18:24
S54	4131	705/26.ccls.	US-PGPUB; USPAT	OR	ON	2005/11/09 18:24
S55	299	S53 and S54	US-PGPUB; USPAT	OR	ON	2005/11/09 18:24
S56	216	(compar\$5 nearshop\$4) and S55	US-PGPUB; USPAT	OR	ON	2005/11/09 18:25
S57	20	(compar\$5 near2 shop\$4) and S55	US-PGPUB; USPAT	OR	ON	2005/11/09 18:28
S58	18	(compar\$5 near2 shop\$4) and S55 and list	US-PGPUB; USPAT	OR	ON	2005/11/09 18:28
S59	1	"6076070".pn.	US-PGPUB; USPAT	OR	ON	2006/05/01 18:38
S60	8	("6076070").URPN.	USPAT	OR	ON	2006/05/02 11:31
S61	10859	(determin\$3 or detect\$3 or check\$3 or find\$3) same (query or RFQ or request or RFP) same (comparison or BOT)	US-PGPUB; USPAT	OR	ON	2006/05/02 11:34
S62	39110	"705"/\$.ccls.	US-PGPUB; USPAT	OR	ON	2006/05/02 11:33
S63	718	S61 and S62	US-PGPUB; USPAT	OR	ON	2006/05/02 11:34

EAST Search History

S64	130	(determin\$3 or detect\$3 or check\$3 or find\$3) same (query or RFQ or request or RFP) same (comparison or BOT) same price	US-PGPUB; USPAT	OR	ON	2006/05/02 11:34
S65	718	S63 and S62	US-PGPUB; USPAT	OR	ON	2006/05/02 11:34
S66	87	S64 and S62	US-PGPUB; USPAT	OR	ON	2006/05/02 11:34

Free

Garg, Yogesh

From: Patel, Jagdish N.
Sent: Tuesday, May 02, 2006 12:48 PM
To: Garg, Yogesh
Subject: search

Dialog NPL Search for 09/676529 Date: 5/2/06

? show files;ds;t s8/9,k/1

File 15:ABI/Inform(R) 1971-2006/May 02
(c) 2006 ProQuest Info&Learning
File 9:Business & Industry(R) Jul/1994-2006/May 01
(c) 2006 The Gale Group
File 810:Business Wire 1986-1999/Feb 28
(c) 1999 Business Wire
File 275:Gale Group Computer DB(TM) 1983-2006/May 01
(c) 2006 The Gale Group
File 476:Financial Times Fulltext 1982-2006/May 03
(c) 2006 Financial Times Ltd
File 610:Business Wire 1999-2006/May 02
(c) 2006 Business Wire.
File 624:McGraw-Hill Publications 1985-2006/May 02
(c) 2006 McGraw-Hill Co. Inc
File 636:Gale Group Newsletter DB(TM) 1987-2006/May 01
(c) 2006 The Gale Group
File 621:Gale Group New Prod. Annou.(R) 1985-2006/May 02
(c) 2006 The Gale Group
File 613:PR Newswire 1999-2006/May 02
(c) 2006 PR Newswire Association Inc
File 813:PR Newswire 1987-1999/Apr 30
(c) 1999 PR Newswire Association Inc
File 16:Gale Group PROMT(R) 1990-2006/May 02
(c) 2006 The Gale Group
File 160:Gale Group PROMT(R) 1972-1989
(c) 1999 The Gale Group
File 634:San Jose Mercury Jun 1985-2006/Apr 30
(c) 2006 San Jose Mercury News
File 148:Gale Group Trade & Industry DB 1976-2006/May 02
(c)2006 The Gale Group
File 20:Dialog Global Reporter 1997-2006/May 02
(c) 2006 Dialog
File 35:Dissertation Abs Online 1861-2006/Apr
(c) 2006 ProQuest Info&Learning
File 583:Gale Group Globalbase(TM) 1986-2002/Dec 13
(c) 2002 The Gale Group

5/2/06

.File 65:Inside Conferences 1993-2006/May 02
(c) 2006 BLDSC all rts. reserv.
File 2:INSPEC 1898-2006/Apr W3
(c) 2006 Institution of Electrical Engineers
File 474:New York Times Abs 1969-2006/May 02
(c) 2006 The New York Times
File 475:Wall Street Journal Abs 1973-2006/May 01
(c) 2006 The New York Times
File 99:Wilson Appl. Sci & Tech Abs 1983-2006/Mar
(c) 2006 The HW Wilson Co.
File 348:EUROPEAN PATENTS 1978-2006/ 200617
(c) 2006 European Patent Office
File 349:PCT FULLTEXT 1979-2006/UB=20060427,UT=20060420
(c) 2006 WIPO/Univentio
File 347:JAPIO Dec 1976-2005/Dec(Updated 060404)
(c) 2006 JPO & JAPIO

Set	Items	Description
S1	1	(ONLINE OR INTERNET OR NETWORK OR WEB) (10N) (COMPARISION - (1W) (SITE OR SHOP OR PROVIDER))
S2	12	(ONLINE OR INTERNET OR NETWORK OR WEB) (10N) (COMPARISION - (1W) (SITE OR SHOP? OR PROVIDER))
S3	23	(ONLINE OR INTERNET OR NETWORK OR WEB) (10N) (COMPARISION - (10N) (SITE OR SHOP? OR PROVIDER))
S4	64128	(DETECT??? ? OR DETERMIN??? ? OR IDENTIFY??? ? OR CHECK??? ? OR FIND??? ?) (5N) (QUERY OR REQUEST OR RFQ OR RFP)
S5	19088	(ONLINE OR INTERNET OR NETWORK OR WEB) (10N) (COMPARISON (- 10N) (SITE OR SHOP? OR PROVIDER))
S6	8	S4 (S) S5
S7	8	RD S6 (unique items)
S8	1	S7 AND PD<20001002

8/9,K/1 (Item 1 from file: 349)
DIALOG(R)File 349:PCT FULLTEXT
(c) 2006 WIPO/Univentio. All rts. reserv.

00441825 **Image available**
METHOD AND APPARATUS FOR ACCESSING ON-LINE STORES
PROCEDE ET APPAREIL PERMETTANT D'ACCEDER A DES BOUTIQUES EN DIRECT

Patent Applicant/Assignee:

THE BOARD OF REGENTS OF THE UNIVERSITY OF WASHINGTON,

Inventor(s):

DOORENBOS Robert B,

ETZIONI Oren,

WELD Daniel S,

Patent and Priority Information (Country, Number, Date):

Patent: WO 9832289 A2 19980723

Application: WO 98US771 19980116 (PCT/WO US9800771)

Priority Application: US 9735623 19970117

Designated States:

(Protection type is "patent" unless otherwise stated - for applications

prior to 2004)

JP AT BE CH DE DK ES FI FR GB GR IE IT LU MC NL PT SE

Main International Patent Class (v7): G06F-017/30

Publication Language: English

Fulltext Word Count: 14116

English Abstract

This invention provides a computer-implemented agent (5) that assists a user (1) in accessing network linked (4 and 6) on-line stores (7). In one aspect, the invention is a method for intelligently routing a user query to on-line stores (7) relevant to that query, extracting relevant data fields from received responses, and intelligently presenting the extracted data in order of estimated interest. In another aspect, the system of this invention implements one or more steps of the method in a centralized or distributed manner on one or more network linked computers (4 and 6). Further, this invention provides a novel heuristically guided process by which the agent is capable of automatically acquiring sufficient information on the characteristics of on-line stores (7) for it to access and shop at those stores.

French Abstract

l'invention concerne un agent informatise qui aide un utilisateur a acceder a des boutiques en direct reliees a un reseau. Un aspect de cette invention concerne un procede qui permet d'acheminer intelligemment une demande d'utilisateur jusqu'a des boutiques en direct propre a la demande; d'extraire les domaines de donnees pertinents a partir des reponses recues et de presenter de maniere intelligente les donnees extraites par ordre d'interet estime. Un autre aspect de cette invention concerne le systeme de l'invention qui met en oeuvre une ou plusieurs etapes du procede de maniere centralisee ou decentralisee sur un ou plusieurs ordinateurs relies a un reseau. Cette invention concerne egalement un nouveau processus guide de maniere heuristique qui permet a l'agent d'acquies automatiquement des informations suffisantes concernant les caracteristiques des boutiques reseautees en direct pour qu'il puisse acceder a ces memes boutiques et y faire ses achats.

Detailed Description

METHOD AND APPARATUS FOR ACCESSING ON-LINE STORES 10- FIELD OF THE INVENTION

The field of this invention relates to information access over networks, and specifically to providing assistance in accessing on-line electronic stores by automatically retrieving product descriptions in response to a user product query.

2a BACKGROUND

The exponential growth of private intranets and the public Internet has produced a daunting labyrinth of increasingly numerous on-line electronic stores and product information databases. Almost any type of product is now available somewhere, but most users cannot find what they

seek, and even expert users waste copious time and effort searching for appropriate on-line stores or other product information sources. One problem is simply the increasingly large number of available sources that are beyond the comprehension of a single user. A second problem, along with this growth in available on-line stores and product information, is a commensurate growth in software utilities and methods to manage, access, and present this information.

Each utility has a different and often unique interface and set of commands and capabilities, and is appropriate for a different set of users and a different set of information types and sources. Thus sheer diversity of available utilities creates problem for users comparable to that created by information explosion. Users are now faced with the twin problems of which tool to use to inquire at which information source.

In the past efforts have been made to provide users with automatic, computer assisted services that can help solve these twin problems of the network revolution. For example, AI researchers have created several prototype software agents that help users with e-mail and netnews filtering (Pattie Maes et al., 1993, Learning interface agents, Proc. QEDINGS of AAAI-93), agents that assist with World Wide Web browsing (H.

Lieberman, 1995, Letizia: An agent that assists web browsing, Proc. 15th Int. Joint-Conf. on A.I., pp. 924-929; Robert 5 Armstrong et al., 1992, Webwatcher: A learning apprentice for the world wide web, Working Notes of the AAAI Spring Symposium: Information Gathering from Heterogeneous, Distributed Environments, pp. 6-12, Stanford University, AAAI Press), agents that schedule meetings (Lisa Dent et al., 10 1992, A personal learning apprentice, Proc. 10th Nat. Conf.

on A.I., pp. 96-103; Pattie Maes, 1994, Agents that reduce on-line electronic stores. Third, existing services and agents have not been easy to adapt to the ever-increasing numbers of stores with their ever-changing response formats.

This is due to the individualized, hand-coded interface to each Internet service and Web site utilized by existing agents (Yigal Arens et al., 1993, Retrieving and integrating data from multiple information sources, International Journal on intelligent and Cooperative Information Systems 2(2):127-158; O. Etzioni et al., 1994, A softbot-based interface to the internet, CACM 37(7):72-75; B. Krulwich, 1995, Bargain finder agent prototype, Technical report, Anderson Consulting; Alon Y. Levy et al., 1995, Data model and query evaluation in global information systems, Journal of intelligent Information Systems, Special issue on Networked

15 information Discovery and Retrieval 5(2); Mike Perkowitz et al., 1995 Category translation: Learning to understand information on the internet, Proc, 15th Int, Joint Conf, on A.I.), Preferably, a service or agent should be able to access a new or changed Internet on-line store in order to automatically learn how to retrieve relevant information from the source,

3* SUMMARY OF THE INVENTION

It is a broad object of this invention to solve these 25 fundamental problems by a method and system that provide a personalized network shopping robot, called a shopbot. A shopbot acts as a user's intelligent assistant by tracking available network product sources or on-line stores, knowing the relevant information and features of each particular 30 product source, and upon user request determining which product sources are relevant to a given query, forwarding the query to the most relevant product sources, understanding the responses returned from each source, and integrating and intelligently presenting the query results to the user.

The shopbots of this invention possess several advantages, including the following. First, a shopbot returns only the relevant product information to the user.

3

on the one hand, each user query is forwarded only to the on line stores in the product domain of interest to a user, On the other hand, responses returned from on-line stores are parsed and understood-so that only the relevant product data 5 items are extracted for user presentation. Duplicate, stale, irrelevant, and mere formatting information items are discarded, Second, a shopbot is fast. Since it automatically searches the relevant on-line stores in parallel, it can present product information as quickly as 10 the fastest primary source returns a response, Despite changing conditions which cause different information sources to fluctuate in speed, a shopbot remains as fast as the fastest on-line store. Stores that have no information to return to a query do not slow the user since the shopbot 15 simply ignores them. Third, shopbots are easily adapted to the ever-increasing number of on-line electronic stores with ever-changing response formats. Shopbots utilize a novel and simple method for describing information sources, A source description is a short and easily understandable collection 20 of strings,

Therefore, in one aspect the invention includes a method for efficient access to product information sources on a network comprising preferably one or more of the following steps: receiving a user query for product information; 25 determining the product information sources or on-line stores in the correct product domain relevant to this query;

retrieving a description of each information source;
formatting a form by which to access these on-line stores
according to the query and to the retrieved description in a
30 manner suitable for each product information source;
transmitting the form to the on-line store; receiving
responses from the product information sources; for each on
line store, understanding and extracting the relevant data
fields according to the retrieved description; and presenting
35 to the user the relevant data from each on-line store in an
intelligent manner ranked by an estimate of its interest to
the user. Advantageously, these steps are performed in

4

parallel to the greatest extent possible. In particular, at
least, all queries are transmitted to all relevant
information sources in parallel without waiting for
intervening responses.

In another aspect, this invention includes a
heuristically guided process by which a shopbot can determine
automatically the description of an on-line store. The
heuristics are collected into domain descriptions for each
product domain to be accessed. The domain description
10 include rules defining typical attributes of products in this
domain and seed knowledge to generate training examples from
an on-line store. This process includes one or more of the
following steps: searching for likely product query forms at
a product information source or on-line store; for each
15 likely query form querying the on-line store both with
products not likely to be carried by the store and also with
popular products likely to be carried; and selecting the form
for future queries which results in the greatest query
success.

In a further aspect the invention comprises a computer
system and apparatus for performing one or more steps of the
method of this invention. The user has a presentation device
attached to a network to which is also attached a plurality
of product information sources or on-line stores. The
25 presentation device receives user queries and displays
shopbot responses. Further, the presentation device performs
one or more of the steps of the method of this invention,
One or more of those steps not performed on this device can
advantageously be performed on network attached shopbot
30 server computers, which respond to functional requests from
the user device. Optionally, the user device can range from
a diskless hand-held terminal, to a PC, to a work station,
and so forth.

5

*** BRIEF DESCRIPTION OF THE DRAWINGS**

These and other features, aspects, and advantages of the
present invention will become better understood by reference

to the accompanying drawings, following description, and 5 appended claims, where.

Fig. 1 illustrates generally a shopbot of this invention;

Fig. 2 illustrates an exemplary user interface of embodiments of the shopbot of Fig. 1;

Fig. 3 illustrates exemplary functional components of the shopbot of Fig. 1;

Fig. 4 illustrates alternative hardware embodiments of the shopbot of Fig. 1;

Fig. 5 illustrates an exemplary product query page for 15 an on-line store;

Fig. 6 illustrates an exemplary product query response page for an on-line store;

Fig. 7 illustrates in general the learning phase of a shopbot of Fig. 1;

Fig. 8 illustrates in more detail the learning phase of □ shopbot of Fig. 1; and

Fig. 9 illustrates in more detail the shopping phase of □ shopbot of Fig. 1,

5* DETAILED DESCRIPTION

For clarity of disclosure, and not by way of limitation, the detailed description of a shopbot of this invention is presented as a method for accessing on-line stores and as a system or apparatus implemented to perform that method.

In the following, first an overview of the invention is presented followed, second, by a detailed discussion of individual components.

6

Selo OVERVIEW OF SHOPBOT ARCHITECTURE,

A shopbot method or system of this invention comprises software and hardware facilities that function together in one or more network attached computers to assist a user to 5 access product information stored in network attached servers (known herein alternatively as "on-line stores," "stores," or "vendors"), Fig. 1 generally illustrates the relationships of a shopbot to a user and to networked on-line stores or product information sources. For example, user 1 accesses 10 user computer 3 through standard interface devices, such as monitor 2. In the course of work, the user needs information from on-line stores 7, attached to the user computer through various network links, such as network links 4 and 6. Since the on-line stores are many, the user can benefit from is assistance in finding needed comparative product information from relevant on-line stores. This assistance is provided by shopbot 5, which maintains an awareness of available on-line stores and their characteristics, and queries them through links 6 on behalf of, or as an agent of, the user.

20 Alternatively, shopbot 5 can partly or wholly reside on user computer 3 or be partially or wholly distributed on the network and accessed by the user through link 4, Groups of on-line stores 7 selling similar sorts of products are grouped into conceptual classes called product 25 domains. For example, one domain can be that of electronic stores for pop/rock music CDs, and another can be that of electronic stores for computer software or hardware products.

In a preferred embodiment illustrated in Fig. 3, shopbot is composed of three major functional modules: user interface 30 36, integrator 37, and I/O manager 41. Briefly, the user interface module interacts with the user to receive user queries for information, and to format and present information responses received from the network attached on line stores, Advantageously, the user interface is adapted 35 to the specific product domain being accessed. In a shopping phase, the integrator module accepts a user product query from the user interface module, formats it for network

7 transmission to each on-line store of the product domain, receives product responses from these stores, understands these responses, and passes the relevant portions of the responses back to the user interface module for display to 5the user, In a learning phase, the integrator module is capable of-accessing a new or changed on-line store and querying it in the process of determining a store description for use during the shopping phase. The I/O manager module performs hardware, operating system, and network specific 10 interfacing for the user interface and integrator modules so that porting a shopbot to different hardware platforms, operating systems, or networks requires only limited changes in well-modularized code.

In particular alternative implementations, either or 15 both of the user interface or the I/O manager modules may be absent. For example, the functions of these modules can be already performed by other operating system components, A shopbot can provide only one or more of the facilities disclosed without providing others, For example, in some 20 embodiments, a shopbot can simply format queries and understand responses, with the on-line store description being externally supplied. In these cases, a learning shopbot can be present elsewhere on the network and can provide vendor description to other shopbots on request, 25 Finally, as is known to those of skill in the art, the functions performed by the described modules may be divided or grouped in alternative fashions among a greater or lesser number of modules, In the absence of specific preferences, the processes of 30 this invention can be implemented in a procedural programming

language, such as C, or an object oriented programming language, such as C++, on the disclosed hardware configurations.

8

The User Interface Module

In more detail, the user interface module has both important functionality that is common to shopbot user interfaces, whatever the product domain to which shopbots are directed and also has adaptations to the particular product domain of a particular shopbot. Turning first to the preferable common functions, one such is the ability to remember a user's preferences for interacting with a shopbot.

Such remembered preferences include, for example, screen display format including preferred product attribute fields and preferred result sort order, the number or identity of on-line stores to query, and so forth.

The user interface module preferably provides one or more windows on the user's screen with several defined views of the query satisfaction process along with certain common user controls, such as screen buttons, for manipulating these windows. In one window, the user interface module presents lists of the on-line stores being consulted with each source symbolically represented as, for example, a network address, an icon, or another compact screen representation. Also displayed is a count of the total number of unique product items received currently. Optionally, clicking on the screen representation of an on-line store opens a further window with either information about this on-line store, or a display of the responses received from it, or access to the on-line store over the network, etc.

In addition to such common functions and controls, a shopbot user interface module preferably implements specific designs, formatting, and fields suitable to the information domain for which it is designed. For example, a shopbot for comparison shopping in a product domain of software stores can have a particular interface presentation containing labeled fields for product name, model, hardware requirements, operating system requirements, price, and so forth. A shopbot for a product domain of pop/rock CD stores can have a particular interface presentation containing

9

labeled fields for artist, group, data, publisher, price, and so forth,

In one shopbot embodiment, most functions and modules reside on network attached servers which a user accesses remotely. For example, the user may access a shopbot over the Internet with the World Wide Web protocols utilizing a

web browser, such as Netscape, In this case the user interface builds HTML formatted pages which are transmitted over the network by the I/O manager. Fig. 2 generally 10 illustrates the user display from an example of such an embodiment, which is further directed to the information domain of on-line, electronic software stores. The shopbot display of Fig. 2 is divided into three sections. Section 11 is a title section generally indicating that this display has 15 results from a shopbot. A shopbot preferably also has a specific input query screen. Section 12 presents the list of on-line stores currently being consulted represented by their WWW addresses 15, which are selectable to provide further information or direct WWW access. At 16 in section 12, those 20 sources which have already returned query results are similarly represented. Section 13 presents the results received so far formatted in accordance with this particular product domain into sections for the major PC operating systems. Each individual item returned, for example item 16, 25 is formatted with product name, price, and an address for the originating on-line store. In this implementation, information display is controlled with the window scrolling and control facilities built into the web browser. This user interface is implemented as a HTML formatted page created at 30 a shopbot server and transmitted to the web browser.

In another embodiment, the functions and modules can reside on the user's local computer. In this case, the shopbot sends queries, receives responses, and formats results locally, The I/O manager utilizes the facilities of 35 the local operating system for user interaction.

Although the user interface is described primarily in terms of windows and buttons, one of skill in the art will 10

recognize that this invention is adaptable to other display paradigms that provide for display of information and input of user commands. For example, the user interface module can control the entire screen and present graphical displays 5 without intervention of a windowing system, The user interface module is preferentially implemented with an object oriented programming language supplemented with a class library providing windowing functions, A preferable implementation uses the Java language together 10 with the java-awt package. See, for example, Flanagan, 1996, Java In A Nutshell, OfReilly & Associates, sections 5 and 19, The Integrator Module Fig. 3 illustrates the preferred functional modules, 15 data bases, and functional interrelationship both of shopbot 30 in general and of integrator module 37 in particular. The integrator preferably consists of three functions: learning phase modules 39, database 40 of product domain and on-line

store descriptions, and shopping phase modules 38, These 20 components are introduced here and described in detail in the following,

In a shopping phase, a comparison shopping query 31 is delivered to the integrator by means of the user interface module 34, The integrator calls the shopping phase modules 25 38 to chose the on-line stores appropriate to the product domain of the query. Next the integrator retrieves the store descriptions for these stores from database 40. Alternately, each shopbot can be specialized to only one product domain, in which case all the store descriptions are retrieved from 30 database 40, In this case, if these are only a small number of stores the store descriptions can alternately be stored in a table in memory. These descriptions of the on-line store and its requirements comprise a set of strings, to be subsequently described in detail, are used by shopping phase 35 modules 38, These modules, first, retrieve product query pages from each on-line store, second, format the user query into fields on each of the pages, and third, then submits the filled-in pages to each store in parallel.

When stores 33 return responses, shopping phase modules 38, using strings in the store description, extract data from 5 the responses and place it into a list of data fields, called a tuple format, relevant to the particular product domain, optionally, each tuple can be assigned a priority order using a method appropriate to the particular user query. Finally when screen display manager of user interface 36 requests 10 data to present to the user, perhaps in response to a more button request, the shopping phase modules pass the tuples to user interface module 34, sorted in priority order if a priority is determined. For example, if the product domain relates to on-line software stores, then the tuples 15 optionally contain such relevant fields as product name, manufacturer, software version number, operating system required, price, etc. An exemplary priority order of the tuples can be by price, by delivery delay, or other factor at user preference. The user display is controlled according to 20 stored user preferences 35,

In a learning phase, the location of a new or changed on-line store is delivered to the integrator. The product domain is externally supplied to the integrator along with the on-line store identification, Alternately, the 25 integrator can call learning phase modules 39 to determine the product domain of the identified store, Next the integrator retrieves the product domain descriptions for the domain from database 40, This domain description includes heuristic rules to be subsequently described which guide the 30 learning phase modules in automatically acquiring store descriptions. Next, the learning phase modules interact with on-line store 33 in a manner tailored by the heuristic rules

from the domain description in order to determine the strings of a vendor description for this on-line store, When a successful vendor description has been determined, it is stored in database 40 for use by the comparison shopping phase modules.

12

The I/O Manager Module

The I/O manager module 41 of Fig. 3 performs hardware, operating system, and network specific interfacing for the integrator module, Network interfacing includes the tasks of sending requests and receiving responses from network linked on-line electronic stores according to protocols recognized by the stores, Since a preferred application of the shopbots of this invention is to shopping on Internet, the I/O manager is responsible for implementing the relevant protocols of the WWW, including TCP/IP, HTTP, and so forth. Optionally, I/O manager 41 can temporarily cache pages and other data in order to improve response time. operating system interfacing includes the task of window management for the user interface module and access to the database services, if present.

Preferably, the I/O manager is constructed from commercially available protocol stacks, windowing libraries, such as the Java.awt package, and other tools. In some implementations, more or less of the I/O manager functions can be performed by other system components on the network attached computer. Optionally, the I/O manager is designed to be scalable to multiple machines, to not require multi threaded or reentrant code, and to be cross platform and persistent,

25 The Shopbot System

The preferred functional structure of a shopbot can be assigned to system hardware components in various alternatives, The preferred alternative in any case depends on which allocation of function achieves a rapid response and reasonable cost, Fig. 4 generally illustrates exemplary shopbot hardware embodiments and options in view of the previous general description. It illustrates the interrelationship of user computer elements 51-56, network 57, on-line stores 58, and shopbot server computers 59-61, Computer 51 is a user computer including a processor, memory, and various attached peripherals, Such peripherals include display device 52, or other device for user interaction,

13

network attachment 54, optional hard disk storage 53, and so forth, Computer 51 can be alternatively a network device without permanent storage, a PC, a work station, or more powerful computer, It is preferred that computer 51 be a PC or a work station running one of the windows operating systems, the Macintosh operating system, or UNIX, Present in

the memory of user computer 51 is, among other software, local shopbot software 55 and local system components 56, The local shopbot software implements one or more of the 10 shopbot functions. The local system components can include, for example, a web browser.

Network@57 can be any network with a plurality of attached on-line stores 58, which can be optionally conceptually classified by type of products sold into a 15 plurality of product domains, In a preferred embodiment, network 5@ is the public Internet or a private intranet supporting the TCP/IP suite of protocols, including such user level protocols as FTP, HTTP, and so forth. The on-line stores are server computers which make their stored product 20 information available using the protocols supported by network 57. Such information can include product type, model, and manufacturers and store price and availability.

In such a network, a shopbot can have various embodiments, In an entirely local embodiment, all shopbot 25 functions reside in local shopbot software 55 on user computer 51, which in this embodiment must have sufficient processing and storage capabilities. In alternative embodiments, one or more of the disclosed shopbot functions can be distributed on other network attached computers.

For example, computer 59 is a store/domain description server for accepting requests for downloading on-line store description or product domain descriptions stored in its database, This database can be stored in memory or on disk using any data management system capable of storing and 35 retrieving compact textual descriptions. Computer 60 is a learning-phase server for performing the computationally more intensive tasks of determining new on-line store descriptions 14

and providing these description either to database, computer 60 or directly to local shopbot 55. Computer 61 is a shopbot server for performing the shopping modules function by accepting user queries and returning search results, perhaps 5 using the facilities of store/domain description server 59 or learning phase server 60, With these network servers, local shopbot software preferably only supports the user interface, which may be performed entirely by a web browser.

Alternately, it can further include the shopping phase 10 modules, which make query routing requests to query server 59 and store description requests to store description server 60. Further, it can include one or both of these latter functions.

The various computers of a shopbot system can be

15 provided with software for performing the methods of this invention either from computer readable media or by loading across a network. This invention is adaptable to known magnetic and optic media, such as disks, tapes and CD-ROM.

5*2e COMPARISON SHOPPING AT ON-LINE STORES

This section describes the regularities of on-line stores that are advantageous to shopbots, and further describes the particular comparison shopping task at which shopbots provide assistance. on-line stores available 25 according to the WWW protocol send HTML formatted documents to a user in order to announce the store, display its products, and receive orders. For a general description of the HTML document description language see, e.g., These store typically are characterized by several 30 regularities in the presentation of their HTML documents. First, their presentation is designed so that customers can find available products and product information quickly.

often, on-line stores provide simple methods to move quickly from the store's home page, that is the document first 35 accessed when a customer visits the store, to a form which a customer can fill out in order to request product information. Fig. 5 illustrates exemplary product 15

information request form 500 adapted to an wide selection of products. By filling in some or all of the available search fields, a customer can search for products. For example, the customer can search for products of a certain category having 5 certain words in their description by selecting a category from field 501 and entering search words in field 501. The customer submits the form to the on-line store by clicking box 503. Line 504 presents "navigation" aids to help the customer access other HTML information documents from the 10 store.

Submission of such a search form causes server computers at the store to search a database of product information describing the on-line store and then return product information to the user, also formatted as one or more HTML 15 documents. On-line stores attempt to create a sense of distinctive identity by using a uniform look and feel for their documents. Although stores differ widely in their product description formats, a particular store advantageously describes all available products in a 20 consistent format. In particular, while different stores use different product description formats, substantially all use vertical separation and white space, that is blank areas of a document, to facilitate customer comprehension. For example, stores start each product descriptions on a separate logical 25 line. Fig. 6 illustrates exemplary product description form

resulting from a query using the words 11 omega jaz. 11 Here, each product with those words in its description is presented on a separate line, such as lines @06, Line 507 is exemplary header information, and line 508 is exemplary trailer 30 information.

On-line vendors respect such regularities because they facilitate comprehension and, thus, sales to human customers. However, these regularities are exploited by a shopbot, Presence of a search form allows a shopbot to 35 simply find product descriptions in a store-independent manner. Regularities in the resulting product description forms permit a shopbot to learn how to access such a store 16

substantially independently of operator assistance,. In particular, these regularities allow the learning procedure to incorporate a strong bias, and thus require only a small number of training examples, In other implementation, a 5shopbot can provide assistance at on-line stores lacking such forms and regularities, However, for such stores substantial operator assistance can be required in order to tailor a shopbot for such a store,

A preferred implementation of shopbots assists users in 10 comparison shopping. In comparison shopping, a customer seeks the on-line store from which to purchase a particular product that is most advantageous according to some criteria.

Therefore, comparison shopping identifies a group of on-line stores that sell the particular product desired by a user, 15 and then ranks the stores in the group based on the customer criteria, e.g., price, speed of delivery,, and so forth. For example, in the domain of computer-products stores, a comparison-shopping shopbot can help answer: "find the lowest price for the Macintosh version of Adobe Photoshop. 11 In 20 general, a shopbot functions according to the following method, Upon receiving such a request, it determines the relevant stores and accesses them in parallel. Next, it searches for the indicated product by retrieving, filling out, and submitting the HTML product search forms available 25 at each on-line store. In a preferred embodiment, parsing and filling out these search forms is done according to a vendor description, which comprise sets of recognition strings, The stores return to the shopbot HTML pages describing their terms for the indicated product. The 30 shopbot parses these returned pages, again preferably, according to the vendor or on-line store descriptions strings. The parsing procedure ignores any header and trailer fields And parses the remaining HTML formatting code into logical lines matching a learned product description 35 format. Returned pages matching a failure template that the shopbot has learned indicates that the search failed at this

on-line store are discarded, Finally, the shopbot sorts the
17
product information, e.g., by ascending order of price, and
generates a summary for the user.

In more detail, the total comparison-shopping problem is
solved in two phases. In a first learning phase described
5 subsequently, a shopbot analyzes on-line stores to learn an
on-line store description, for handling HTML pages from the
site. This phase is more computationally expensive, but is
performed in advance of actual customer comparison shopping,
and needs to be done only once per store. However, if a
10 vendor "remodels" the store with different HTML formatting,
providing different search forms or different product
description page formats, then this first learning phase is
repeated for that vendor. In a second shopping phase, which
is less computationally expensive, the learned information is
15 actually used by a customer for comparison shopping. The
shopping phase is implemented according to the previously
p
described shopbot architecture for rapid parallel access of
relevant on-line stores.

In particular, a shopbot's implementation and graphical
20 user interface advantageously utilizes certain important
principles. First, the shopbot in the comparison shopping
phase is fast. Because most of the computational work has
been in advance of shopping in the learning phase, the
shopping phase can be fast. In fact, although the most time
25 consuming step is fetching pages over the network, since such
fetching is done in parallel across all vendors, a shopbot is
faster than an expert human. Second, a shopbot provides its
user with continual feedback informing the user which vendors
are being contacted and what prices have been found so far
30 and permitting user interrupts at any time. Third, the
shopbot provides the user with enough context around any
information it extracts so that the user can verify its
conclusion or investigate manually. For the user's
convenience, ShopBot indicates the store's home page, the
35 search form it used, and each full product description found.

In summary, Table 1 outlines the input and output to a
comparison shopping task,

18

TABLE 1 - COMPARISON SHOPPING
Given.

le A description on the particular on-line shopping
domain, including information about product
attributes re,g., name, manufacturerl pr ce, and
so forth, useful for discriminating between
different products and between variants of the

same product and typical popular products;
2a URL's for the home pages of possible on-line stores;
3* An attribute A, e.g., the price, by which the user wants to compare vendors; and
4e A specification of the desired product in terms of the values of selected attributes, e.g., name is Photoshop,
Determine.

5* The set of on-line stores where the desired product is available, sorted by A,
Generally, input to the learning phase is items 1 and 2.

Using this information and a constrained learning process, a shopbot is capable of learning on-line store descriptions for accessing product information in the on-line stores. Items 3 and 4 are input to the shopping phase. A user inputs 20 attributes of the products of interest in a manner consistent with the domain, and the shopbot retrieves product -information from the relevant on-line stores.

Although, this description of a shopbot is directed to the comparison shopping task, it will be apparent to those of skill in the art that a shopbot can equally well be constructed for other general shopping tasks involving network search. Further, for tasks in which the relevant on line stores present significant regularities, a learning phase can be constructed which learns the relevant on-line store descriptions,

5*3* LEARNING ON-LINE STORE DESCRIPTIONS

This subsection describes the first learning phase of shopbot processing in which a shopbot learns the information necessary for it to assist a user in the second comparison shopping phase which can be repeatedly performed as long as

the learning process preferably proceeds with a minimum of training examples and in an unsupervised manner. To minimize disruption of an on-line store's activities, it is preferable that shopbot learning retrieve a strictly limited number of HTML documents from the store. Further, it is preferable that operator intervention be minimized or eliminated in order to a shopbot to be able to comparison-shop at a new or revised on-line store. In this subsection, first, the processing, input and output of the learning process are described in general, followed, second, by details of each major step.

In general, a shopbot can learn a vendor description for an on-line store by using heuristics which exploit the regularities typically present in an on-line store's HTML

documents. These heuristics strongly direct the learning process according to these regularities. First, these heuristics assume that every product description is somehow vertical-space-delimited, by, e.g., starting a paragraph, a new row in a table, a new line, and so forth. Such vertical space-delimitation is specified by HTML tags such as <p>, <tr>, , or
. Accordingly, after removing header and trailer information, the heuristics divide remaining HTML code of each page into "logical lines" representing groups of vertical-space-delimited text. Second, the heuristics assume that every product is described in the same format.

Accordingly, each "logical line" that is found is abstracted into a "line description" by removing the arguments from HTML tags and replacing all occurrences of intervening text with the variable "text." The most successful such line description is used to describe that on-line store's product description pages. In fact this last regularity is expected since most on-line stores retrieve product information from a relational database with a program to create a custom information page in a simple format. In more detail, the learning process is product domain independent. All domain dependence is input to the learning process as data contained in a domain description.

20

Therefore, in order to shop in a new product domain whose on-line stores share described regularities, the learning modules merely need to fetch the appropriate domain description from storage. A domain description contains three categories of information: a description of the product attributes, heuristics for understanding on-line store pages, and seed knowledge to bootstrap learning. Product attributes are categories relevant for describing products in this domain. For example, for a computer software domain, product attributes can include product name, manufacturer, price, hardware requirements, operating system requirements, and so forth. Heuristics for understanding on-line store pages recognize the terminology used in the search. These heuristics are preferably of the form of rules whose antecedents match typical words used to describe input fields on query forms and whose consequents specify which product query attributes to fill into the input fields. For example, turning to Fig. 5, for search forms such heuristics can recognize the words "product category" in field 501 or "description" in field 502 as possibly identifying input fields for product attributes or description, respectively. Finally, sample seed knowledge is used as test queries to begin learning of on-line store product description pages.

This knowledge includes products almost certainly not in the

25 domain in order to learn to recognize search failure pages.

It also includes common products likely to be present in many stores in order to learn successful product description pages. For example, for computer software seed knowledge can include the products "Microsoft Encarta.11 "Abode Photoshop", In addition, although this detailed description of the learning phase modules of this invention is directed toward their application toward learning descriptions of on-line stores, the methods of these modules are not so limited, They are equally applicable to any information source that 35 obeys the previously mentioned format regularities, In particular, they are applicable to learning how to access any information source that has an information search form, that 21

returns failure pages upon information search failure, that returns information pages upon search success having a uniform format in which information items are separated by recognizable formatting codes, such as HTML codes for 5 vertical white space.

Summary Of The Learning Process

On-line stores typically provide a search form that a user can fill in with, e.g., the name and manufacturer of a desired product and then submit, e.g., by clicking on a "submit" or "search" button. The on-line store responds by returning a page containing product information in a consistent HTML format for user scrutiny. Thus, for comparison-shopping, a shopbot accesses the page containing the search form, properly fills in the accessed search form, and then 'extracts relevant product information from the returned page(s). For the latter step, the format of information on a product information page needs to be represented,

Therefore, the shopbot learning problem is comprised of three sub-problems. A first sub-problem is that of finding the correct product search form for that on-line store. Some on-line stores have several search forms, only one of which can be used to locate product information, A second sub problem is to determine how to fill in the correct product search form, that is what product attributes, e.g., product name and manufacturer, to enter into which fields in the search form. Finally, a third subproblem is to learn how to extract the product information from the information pages returned from the search.

Solutions to these sub-problems are interdependent. A shopbot in a learning phase (hereinafter called a "learner") cannot be certain that a particular search form is appropriate until it knows how to fill it in and how to 35 understand returned results. Therefore, the learning-phase

shopbot searches combinations of solutions to these three sub-problems in order to pick the best combination.

22

Fig. 7 illustrates the general process for a learning phase shopbot. Starting with URL 551 for the home page of a particular on-line store, at step 552 the learner searches at that store for candidate HTML product search forms. It thereby determines limited set 553 of candidate search forms, F , for further testing. At step 554, the learner tests each form F_i to compute an estimate E_j for how successful the comparison-shopping phase would be if form F_j were chosen by the learner. To compute the estimate, the learner determines 10 attribute mappings directing how to fill in the fields of the form, and then makes several "test queries," using the form to search both for several popular products of the shopping domain and also for products certain not to be in the shopping domain. The results of these test queries provide, is first, training examples from which the learner determines the format of product descriptions in the pages resulting from form F_i including the header, trailer, and item format strings. Second, test queries provide examples of search failure pages from which the learner determines the failure 20 string. Third, the results are also used to compute E_j . An estimate of the learner's success in extracting information for these popular products provides an estimate of how well the system would do in general for ordinary products. Thus, the learner determines complete vendor descriptions 555 based 25 on each of the candidate search forms found. Finally at step 556, the learner picks the form with the best estimate, E_j .

The learner's final output 557 consists of a vendor description: the chosen form's URL, the failure string, the attribute mappings, the header and trailer strings, and the 30 item format.

In the following, key steps of this process are described in greater detail. These steps depend on various heuristics. This invention is equally adaptable to alternative heuristics that achieve similar functions in 35 similar manners.

Finding Candidate Forms

23

The first step is to find candidate search forms at the store's web site. Forms are parts of HTML formatted web pages starting with a `H<form>` tag and ending with a `If</form>` tag. An exhaustive search, starting at the store's 5 home page and recursively following all HTML links, is much less preferred. Such a search could, for some stores, eventually reach most of the millions of pages on the WWW,

and, second, could result in a large number of pages being fetched from the given store, which would place a heavy burden on that store's server,
To avoid these problems, it is preferred to place a limit, currently preferably between 25 and 100 and most preferably 50, on the number of pages a learner fetches while trying to find candidate forms. Given such a limit, however, the search procedure is preferably more selective. For example, if a site has 500 pages, only one of which has the right form on it, but a learner looks at only 50 of them, then a random search would have only a 10% chance of success. Therefore the learner incorporates heuristic techniques designed to increase its chances of finding the right page despite a limited search.

According to this technique, the learner prioritizes fetching of pages according to a priority scoring function.

Lower scores are better, i.e., considered more likely to be the page containing the right search form. A priority queue of URLs of pages to be fetched is maintained. Initially, this queue contains just the URL of the store's home page with score 0. The learner repeatedly removes the highest priority, or lowest scored, URL from the queue, fetches that page, and adds to the queue the URL's from any links contained in that page. As soon as 50 pages have been fetched, the search stops, and the learner searches for forms on the 50 pages retrieved,
No page gets fetched twice By keeping track of which pages have already been fetched. Further, the learner also has a list of forbidden domains from which it never fetches pages. Forbidden domains are URLs that many WWW sites are linked to, e.g., netscape.com, microsoft.com, lycos.com, altavista.digital.com, and so forth, that are known not to be part of any on-line store's WWW site.

The following pseudocode illustrates the process for finding a set of candidate forms,
PROCEDURE FIND -CANDIDATE-FORMS (starting url, max pages).

```
/* Initial queue is just starting url with score 0 */
initialize priority queue: pages-to-do = <starting-url,
priority = 0>
initialize set page
already
done = empty set;
initialize set forms found = empty set;
/* main loop: fetch a page, add its links to queue
```

```

WHILE ( (pages
to-do is not empty) AND
(size(pages
already
done) < max
pages) ) DO BEGIN
remove the minimum score item from pages-to-do queue;
name this item l1<url, this score>;
IF ( (url is in pages
already
done) OR (url accesses a
FORBIDDEN DOMAIN) ) THEN skip this url go on to the
next iteration of the WHILE loop;
fetch the url; name the HTML text fetched "page";
add url to pages-already
done;
FOR EACH form on page, add that form to forms - found
FOR EACH outgoing HTML link on page DO BEGIN
LET u-link denote the url of the link and t-link
denote the text of the link; i.e., the HTML
code for the link has the form "<a href
l1 u
link l1> t-link </a>l1";
IF (t
link contains as a substring the words
"search,," "find," or a variant thereof) THEN
score = this score + 0,1
ELSE IF (t-link contains a substring "text mode,l1
"text only mode,l1 "no graphics," or a variant
thereof)
THEN score = this-score + 0,2;
ELSE BEGIN
25
W.O 98/32289 PCTfUS98/00771
/* see explanation of "position-component" in
text */
score = this-score + 1.0 + position-component;
IF (u
link is in a different domain than url)
THEN score = score + 2.0;
END;
add <u-link.score> to the priority queue
pages
to-do
END /* of FOR loop
END /* of WHILE loop
RETURN forms found
END /* of PROCEDURE FIND-CANDIDATE-FORMS
The scoring function is designed to give priority to
those URLs considered more likely to contain or to lead to
the desired search form, Five heuristic rules are used to

```

compute the score. First, the score for a given link is always higher than the score from the page containing it.

The score is always computed by adding a positive number to a previous score. This is because the farther away from the store's home page, i.e., the more links followed from the home page, the less likely is the search form. On-line stores usually put the search form within 2 or 3 clicks from their home page. Second, a link that contains the word "search" or "find" is likely to lead to the search form page, so it gets a relatively good score. Third, a link that points to a "text-only model" section of the store's web site gets a good score, since a shopbot is more likely to understand text pages. Fourth, if there are multiple links on a page, the links near the beginning and end of the page are more likely to lead to a search form than links in the middle of the page. This is because many pages contain long lists of links to different sections of the store; a link to a general search form page is unlikely to occur in the middle of such a list. To "penalize" links in the middle of a page and "reward" links at the beginning or end, a "position component" is added to the score. The position-component is 26

for the first and last link on the page; 1 for the second and next-to-last link on the page; 2 for the third and third from-last link on the page; and so on. Finally, fifth, if a link jumps to a new domain, i.e., a page from "foo.bar.com" has a link to "aaa.bbb.com". then the link is penalized, since it is likely to lead to some completely different site that isn't part of the store. However, such links are not discarded because some stores' WWW sites do in fact span multiple domains.

Filling out A Candidate Form

Fig. 8 illustrates in greater detail the process of step 554 of Fig. 7. In general, starting with a set of candidate forms, forms 553 of Fig. 7, at step 602 a learner selects 15 from the candidate forms those likely to be product search forms. At step 603, a learner then uses rules from the domain description to build an attribute mapping, which guides the learner in filling in form fields appropriately. Using these as input, at step 605 a learner queries the on line store with dummy products in order to determine a failure format string, and at step 607, it queries the store with popular products in order to determine the format of product information in a successful product page. If any of these step fail, a learner abandons processing the current 25 candidate form and starts with the next available form, if any. once all candidate forms have been processed or abandoned, that form having the highest evaluation, along with the determined descriptive strings, is used for the description of this on-line store.

This process is described in more detail in the following paragraphs. First, starting with-URL 601 giving the location of the next candidate form, a learner then determines how to fill in that candidate form, that is what product attributes, e.g., product name, manufacturer, and so forth, to enter into each of the fields in the search form, To determine this, a learner first examines the HTML text of the form to extract the various type-in input fields, which

27 are specified by HTML <input> tags, and locates the corresponding prompts, i.e., the text immediately preceding the input field which normally prompts the user concerning what data should be entered in that field, Next, a learner checks for the presence of several conditions-which indicate that the form is almost certainly not a product search form, In this case, the form is discarded at step 609 without an attempt to fill it out.

Four such preferable conditions are described here, A first is such condition is that submitting the form would require accessing a forbidden domain. A second such condition is that one of the type-in input fields is of type PASSWORD or TEXTAREA, as specified in attributes of the HTML tags. Such kinds of input fields are rarely found in search forms. A third such condition is that the form contains no input fields of type TEXT, i.e., no ordinary type-in input fields, at all. A fourth such condition is that one of the field's prompts contains such words as "mastercard," "email," "email." "phone," "telephone." and so forth. In this case, the form is probably a user registration form or an order form, and not a search form. other such conditions can be used as appropriate in various product domains or as on-line stores evolve and change.

If none of these conditions hold, then a learner proceeds to step 603 in order to determine which product attributes, e.g., name, manufacturer, and so forth, are to be entered into each type-in input field, This determination is made differently for different product domains, i.e., for computer software stores there is one set of attributes to choose from, such as product name, version, hardware platform, operating system, and so forth, while for music CD's there are different attributes, such as artist, album title, and so forth, In general, the determination is made using a product-domain dependent set of rules that test the prompt of a field and decide what product attributes to enter into that field. An exemplary rule might test the field's prompt for the word "name." and if found,, fill in that field

28 with the product name. A further exemplary rule for a

product domain without part numbers, might test the field's prompt for the words "part number," and if found, leave that field blank, since the learner knows nothing about part numbers. For each product domain, e.g., software, CD's, etc., the domain description must provide a learner with a list of such rules. Then, for each input field in a candidate form, a learner system sequences through this list of rules and applies the first rule whose test matches the field's prompt. If a rule applies, an attribute mapping pair is added to the already found mapping pairs. The pair comprises the string in the field prompt that was matched by the rule paired with the name of the product attribute that the rule indicates should be entered in this fill-in-field. If none applies, the field is not filled in with anything.

Output 604 from step 603 includes the URL of the candidate form together with the constructed attribute mapping.

20 Learning Result Formats

Having determined the attribute mappings, which guide how to fill in a candidate form, at steps 605 and 607 a shopbot in a learning phase determines how to extract information! in other words to parse, result pages returned from the on-line store after this filled-in form has been submitted. Preferably, the learner relies on several typical regularities of the product information pages. First, for each form, the result pages typically are of two types: a "failure" type, when nothing in the store's database matched the query parameters; and a "success" type, when one or more items matched the query parameters. Second, success pages typically consist of a header, a body, and a tailer, where the header and tailer are consistent across pages from different product searches, and where the body contains all the desired product information, along with possibly irrelevant information as well. Third, the product descriptions typically have the same unique format, not

W.O 98/32289 PCT/US98/00771

possessed by anything else in the body of the page,. Using these regularities, in order to parse a result page a learner solves three additional sub-problems: first, learning the generalized failure template; second, learning to remove irrelevant header and tailer information; and third, learning product description formats,

Accordingly, at step 605 a shopbot first determines a general failure template for a form by querying the form with several "dummy" products almost certainly not in the database, such as the product named llqrsabcdummynosuchprod" from company "MadeUpManufacturerName," More specifically,, the result page for one dummy product, e.g.,

llqrsabcdummynosuchprod,11 is fetched and each occurrence of that dummy product name in the result page is replaced with a 15 special placeholder, i.e., replace each occurrence of the string llqrsabcdummynosuchprod" in the result page with the string "****DUMMY-NAME****", and replace each occurrence of "MadeUpManufacturerName11 with "****DUMMY-MANUFACTURER****".

This procedure is repeated for several more such dummy 20 products with different names. If the result pages, after replacement of strings, are the same in every case, then the learner records this page as the failure format string for this form. If the different result pages for different dummy queries are not the same, then the learner abandons further 25 processing with this candidate form at step 609 and goes on to the next candidate form found in set 553 of Fig. 7, Output 606 from this step includes the URL of the form, attribute mappings for the form, and the failure format string by which search failures can be recognized. The 30 following pseudocode illustrates the procedure for querying with dummy products.

```

initialize result
pages = empty-set;
FOR EACH ( dummy product in this domain denoted by
<attr1=value1 , attr2=value2 , , , j, attrN=valueN> ) DO BEGIN
page = (fill out this form using this dummy product
and fetch the result page);
30
FOR i=1 TO N DO BEGIN
replace all occurrences of "valuej" in page
with "****DUMMY-attri****";
END;
add page to the set result
pages;
END;
IF (all elements in result
pages are the same)
THEN FAILURE-FORMAT = this one page
ELSE skip this form and go to the next form from
step 1;
Next, at step 607 in order to learn how to recognize
pages returned from a successful product query, a shopbot
learner queries the form with several popular products from
15 the domain as provided in the domain description, e.g., the
current best-selling products in this domain. It compares
each result page for these products against the failure
format learned above; any page that matches the failure
format is assumed to represent a failed search for this form
20 and is discarded at step 609. If the majority of the test
queries with the popular products are failures rather than
successes, the learner determines that this is not the

```

appropriate search form to use for the vendor, and it goes on to the next candidate form found in set 553. otherwise, the 25 learner records generalized templates for the header and trailer of success pages, by replacing words which are product attributes with fixed standard strings, and then by finding the longest matching prefixes and suffixes substrings of the success pages obtained from the test queries.

The output of this process now includes the header and trailer strings by which the uninformative portions of a successful product query page can be recognized and discarded. The following pseudocode illustrates this process.

```
initialize result
pages = empty-set;
initialize result
copies = empty-set;
31
98/32289 PCT[US98/00771
FOR EACH ( test product in this domain denoted by
<attr1=value1, attr2=value2 . . . I I I attrN=valueN> ) DO BEGIN
page = (fill out this form using this dummy product
and fetch the result page);
copy
of
page = page;
FOR i=1 TO N DO BEGIN
replace all occurrences of "valuei" in
copy
of-page with "****DUMMY-attri****"
END;
IF ( copy
of-page is different from FAILURE-FORMAT)
THEN add page to the set result-pages;
END;
IF ( size(result
pages) < the number of test products)
THEN give up on this form and go on to the next
is form from step 1;
set HEADER = longest common prefix substring of all the
result
copies;
set TAILER = longest common suffix substring of all the
result
copies;
Continuing in step 607, a learner now uses the bodies of
the pages from successful searches as training examples from
which to determined the format of product descriptions in the
result pages for this form, Each such page contains one or
25 more product descriptions, each containing information about
```

a particular product, or version of a product, that matched the query parameters. The format of these product descriptions varies widely across vendors. However, at each particular vendor, all the product descriptions usually have 30 the same abstract format. The learning phase processing searches through the possible abstract formats and picks the best one, i.e., the one it determines to be most likely to correspond to product descriptions at this site, The abstract formats are described by strings of HTML 35 tags together with keyword "text". The abstract form of a fragment of HTML is obtained by removing the arguments from HTML tags and replacing all occurrences of intervening text 32

with the keyword "text," For example, the HTML source string "Clickherefor Encarta.11 is abstracted into the abstract form string $\Pitext<a>texttext,\Pi$

There are many abstract formats to consider in the search, i.e., many possible sequences of HTML tag names and "text". Even considering only the finitely many which actually occur in one of the bodies of the success pages from the test products, there is still a large number of possible formats to consider. So this number is further reduced by assuming that every product description starts on a fresh line, as specified by certain HTML tags such as <p>,
, , etc. Therefore, the process first breaks the body of each result page into logical lines which are strings separated by vertical-space-delimited HTML tags, and only then considers abstract formats that correspond to at least one of the logical lines in one of the result pages.

The bodies of success pages typically contain logical lines with a wide variety of abstract formats, only one of which corresponds to product descriptions. The learner uses a heuristic ranking process to choose which format is most likely to be the one the store uses for product descriptions.

(A preferred ranking function is the sum of the number of logical lines of that format in which some text, not just white space, was found, plus the number of logical lines of that format in which a price was found, plus the number of logical lines in which one or more of the required attributes were found. This heuristic exploits the fact that since the test queries are for popular products, on-line stores tend to stock multiple versions of each product, leading to a plurality of product descriptions on a successful page. This ranking function reflects both the number of popular products that were found and the amount of information present about each one. The exact details of the heuristic ranking 35 function do not appear to be crucial, since there is typically a large disparity between the rankings of the

"right" format and alternative "wrong" formats. This

33

invention is adaptable to other heuristic ranking functions that achieve similar discriminations.

Final output 608 of step 607 includes all the components of an on-line store description built from the current 5 candidate form as well as the value of the ranking function.

If the form is to be abandoned as described for previous steps, this ranking is set to a large negative value so that it will be ignored in the further processing. This last process of step 607 is illustrated in the following 10 pseudocode.

```
lines = empty set;
formats = empty set;
FOR EACH page in result
  pages DO BEGIN
    Let body be the substring of page remaining after
    removing the prefix and suffix substrings
    matching header and tailer;
    Divide the body string into a set of logical lines,
    or substrings, at occurrences of vertical
    space-delimiting HTML tags. such as <p>, <br>,
    etc;
    FOR EACH logical line DO BEGIN
      Add this line to the set lines;
      Add abstract version of this line to the set
      formats;
    END;
  END;
RETURN (the format in formats which maximizes
evaluate ranking function)
FUNCTION evaluate (format).
```

```
INTEGER n = 0;
BEGIN
FOR EACH line in lines FOR WHICH (abstract
version of line)=format DO BEGIN '
IF (line contains any text other than
white space) THEN n=n + 1;
34
IF (line contains a price, i.e., a dollar
sign followed by digits) THEN n=n +
1;
IF (line contains any of the test product
valuei"s as a substring) THEN n=n +
END;
RETURN (n);
END
```

Generating the Vendor Description

Finally, a learner must choose the candidate form to use for this on-line store. Returning to Fig. 7, the learner has processed each candidate form found in set 553 according to 15 the previously described steps illustrated in Fig. 8. For each form¹ it determines how to fill in the query attributes, and how to parse the result pages to find a best abstract format, that is the format that maximizes the function $evaluate(format)$. At step 556, it chooses one of those forms 20 with the greatest value of the ranking function for shopping at this on-line store. As mentioned above, this choice is based on making an estimate $E_i f$ for each form F_i of how successful the shopping would be if form F_i were chosen by the learner. The E_i used is the value of the evaluate function 25 for the winning abstract product description format. This function reflects both the number of the popular products that were found and the amount of information present about each one. Thus the form selected is the one whose best format has the greatest value of evaluate among all other 30 candidate forms, and thus the greatest use in accessing this on-line store.

once the learning phase has chosen a form, it records a vendor description (See Table 2) for future use by the shopping phase. If the learner is unable to find any form 35 that yields a successful search on a majority of the popular products, then shopbot abandons this vendor.

35

The learning phase runs once per merchant prior to any shopping at this merchant. The learner's running time is linear in the number of vendors, the number of forms at a vendor's site, the number of "test queries," and the number 5 of lines on the result page. The learner typically takes 515 minutes-per vendor.

5040 SHOPPING AT ON-LINE STORES

TABLE 2 - A VENDOR DESCRIPTION

The URL of a page containing a product search form, and optionally an encoding of the preferred search form on this page

Pairs of strings, each pair being an attribute name and a field name, for mapping of product query attributes to fields of that form,

Strings used by parsing functions for extracting product data from pages returned from the on-line store by matching portions of the returned pages, including

A string that matches a unique portion of a search failure page (e.g., "Product not found"),

Header and trailer strings that match mere

header and trailer formatting information used in search success pages.

An abstract format string that matches the components of product descriptions in search success pages and is used to extract information from those descriptions.

The output of the first learning phase is an on-line store description, which together with the domain description is used in the second comparison-shopping phase. Table 2 lists the preferred components of an on-line store description. In a preferred embodiment, the output can be several strings used by the functions subsequently described for comparison shopping. These strings include location string, attribute mappings, failure string, header and trailer strings, and an item format string. The location string contains the URL of the WWW page containing the product search form. If there are multiple forms on the search page, the location string additionally encodes which

36 form to use for product searching. The attribute mappings are pairs of strings of the form ("attribute-name", "field name"). The "attribute names" are names of the attributes defined in the domain-description, e.g., "title", "manufacturer", "artist" for a CD store domain. The "field names" are labels for the fields on the search form to be filled in with values for the corresponding "attribute name," HTML requires a label for each fill-in input field. The failure format is a single string matching a distinctive 10 portion of the search failure page returned from the on-line store if a product search fails. The header and trailer strings match the header and trailer formatting portions of a successful product search page returned from the on-line store. Finally, the item format is a single string which 15 matches an abstract version of each "logical line" of product information returned on a successful product search page. An abstract version of a logical line has all text, that is everything other than HTML keywords, replaced by the string "text," This invention is adaptable to other formats for 20 vendor descriptions. For example, it is adaptable to a language based on regular expressions for parsing HTML formatted documents.

During the second comparison-shopping phase, the vendor and domain descriptions are used to assist a user in 25 comparison-shopping at an on-line electronic store on the Internet. In a preferred embodiment in which the vendor description contains the above described strings, this phase proceeds generally as illustrated in Fig. 9, By means of user interface 650, a user inputs a product information

30 request to a shopbot, which can be, for example, purchase
 request 655 to find the cheapest price for a product,
 Shopping phase modules 651 proceed generally according to the
 process illustrated in Fig. 9. Domain description 653 for
 the product query is retrieved from the store/domain
 35 description database, The user request is represented as the
 values of the product attributes appropriate to this domain
 as identified in the domain description. Next, shopbot 651
 37
 retrieves the vendor descriptions for relevant on-line stores
 having products in this domain. Guided by vendor
 descriptions 654, for each vendor, shopbot 651 access that
 on-line stores's product search form, fills in the form with
 5the query attributes, and submits filled-in form 656 to the
 on-line store, The store returns product information result
 pages 657 from which shopbot 651 parses responsive product
 information. These results are presented to the user by user
 interface 650 as, for example, best buys 658 for the product
 10 of interest. In more detail shopbot 651 process according to
 the following pseudocode, The input to this procedure are
 values of the attributes supplied by the user which defines
 the user's current comparison-shopping request.

```

PROCEDURE shop
for-item (<attrj=valuej , attr2=value2,
attrn=valuen>)
BEGIN
Initialize the set results = empty set;
form = fetched HTML page specified by the location
string;
Let "mapping" denote the learned attribute-to-field
mapping;
FOR EACH input field with name f in form DO BEGIN
Fill in field f with the provided value of the
attribute whose name corresponds to f in
the attribute mapping pairs
END;
Submit the filled-in form and get the result-page;
temp = result
page;
FOR i=1 TO N DO BEGIN
replace all occurrences of "valuej" in temp
with "***Dummy-attrj***"
END;
/*search failed, return empty set for results
IF temp = the learned failure format THEN EXIT
Delete the first (length of learned header)
characters of page;
38
Delete the initial and terminal portions of
result

```

page matching the header and tailer
 strings, respectively;
 Divide the remaining result
 page into a set of
 logical lines (substrings) by separated by
 occurrences of <p>,
, etc;
 FOR EACH of these logical line DO BEGIN
 Make abstract version of logical line by
 replacing everything other than HTML
 keywords with the string "text";
 IF the abstract = the learned item format
 string THEN add this-line to results
 RETURN results
 END; /* shop
 Immediately after a page is fetched and before any
 further processing, either in the learning or shopping
 phases, it is checked for any graphics that can be replaced
 with text, Such replaceable graphics is specified in HTML by
 using an tag with an "ALT" argument, which gives the
 text replacement. In other words the HTML keyword <IMG ...

ALT=11some text here" ... > is replaced by just "some text
 here."

The shopbot stores vendor descriptions, which are a
 repository of knowledge about on-line stores accessible on
 the Internet, in a description database. Since a comparison
 shopper uses the shopper to comparison shop at an on-line
 store only after the learning phase has been completed at
 that store, a shopbot is able to immediately access on-line
 30 stores and quickly search for a desired product. The shopbot
 shopper is both methodical and effective at actually finding
 products at a given vendor,

6* EXAMPLES

The invention is further described in the following
 example which is in no way intended to limit the scope of the
 invention.

39

SHOPBOT TRIALS

Evaluating ShopBot UtilitV

In a first experiment, shopbot usefulness was measured
 by comparison with manual comparison shopping. seven
 5subjects were chosen who were novices at electronic shopping
 but who did have experience using Netscape Navigator, The
 subjects were divided into three groups.

Three subjects who used ShopBot;

2, Two subjects who used Netscape Navigator's search tools
 and who were given the URLs of twelve software stores
 used by shopbot; and

3, Two subjects who were limited to Netscape Navigator's
 search tools,

Two independent parties suggested four popular software products for which to comparison shop. The products were Netscape Navigator, Hummingbird eXceed for Windows, Microsoft

TABLE 3 - SHOPBOT TRIALS

Group Time Navig eXceed Word Quicken

(min:sec) ator

13:20 \$30,71 \$373,06 \$282,71 \$42.95

2 112:30 38,21 (not found) 282,71 4 1. 50

58:30 40,95 610.00 294,97 42,95

The shopbot group completed its search task much faster than the other subjects, and generally found prices at least as low as found by the other groups. The group 3 subjects 35 limited to Netscape Navigator's search methods never found a lower price than ShopBot users. Providing the list of s tore

40

W.O 98/32289 PCT[US98/00771

URLs actually slowed the subjects down. For example, one group 2 subject failed to find a price for eXceed, and the other found a low price on an inappropriate version. These trials demonstrate a shopbot's utility for comparison shopping.

Acquisition of New Software Vendors

To assess the generality of the shopbot learning phase, an independent party not familiar with shopbot processes 10 found ten on-line stores that sell popular software products and that have a search index at their WWW site. With the heuristics of the preferred embodiment, a shopbot was able to learn how to comparison shop at all ten vendors. Shopbot currently shops at twelve software vendors, the 15 aforementioned ten plus two more that were used to guide the original design. This demonstrates the generality of shopbot's architecture and learning processes and heuristics within the software domain.

In more detail, Table 4 shows the line descriptions and 20 heuristic rankings found during learning product description formats for two software vendors. In both cases, a shopbot picked the correct line description corresponding to product descriptions. Other vendors with different product formats have also been consistently learned,

41

.WO 98/32289 PCTIUS98/00771

TABLE 4 - FORMATS AND RANK

Internet Shopping Network NECX Direct

(http://w,@iw,internet,net) (http://necxdirect.necx.com)

Line Description Rank Line Description Rank

<a>text<jta>text 324 <a>texttext 402

(correct format)

(correct format)

<a>texttext 41 <h4>text<a>text 21,

<h2>text<a>text 3 <a>texttext 12
text
</h2>text 0 <a>text 4

text 0 , br> 0
</h4> 0
text 0

Generality Across Product Domains

To date, a shopbot has been tested in the domains of CDfs as well as software products. A domain definition that enables a shopbot to shop at pop/rock on-line CD stores has been defined. The CD domain was chosen to demonstrate the versatility and scope of shopbot's learning processes. With one day of work on describing the CD domain, a shopbot was able to shop successfully at four CD stores.

7* SPECIFIC EMBODIMENTS, CITATION OF REFERENCES

The present invention is not to be limited in scope by the specific embodiments described herein. Indeed, various modifications of the invention in addition to those described herein will become apparent to those skilled in the art from the foregoing description and accompanying figures. Such modifications are intended to fall within the scope of the appended claims,

Various publications are cited herein, the disclosures of which are incorporated by reference in their entireties.

42

Claim

An apparatus to assist a user in querying for
5 information about a product at one or more on-line electronic stores, said apparatus comprising:

a, a first memory means for storing product domain descriptions;

b, a second memory means for storing descriptions of
10 said on-line stores;

C, a first processor means for: (i) receiving a user query for said product; (ii) fetching a product domain description for the domain of said product and for fetching on-line store descriptions of one of more on-line stores for
15 said product domain; (iii) accessing a product query form at said on-line stores according to said on-line store descriptions; (iv) filling-in said product query forms according to said domain description and said on-line store descriptions and for submitting said filled-in query forms to
20 said on-line stores; (v) receiving response pages from said on-line stores; (vi) extracting product information from said response pages according to said on-line store descriptions; and (vii) presenting said extracted information to said user.

25 2, The apparatus of claim 1 wherein said product domain is the domain of computer software and hardware products or the

domain of music CDs.

3 The apparatus of claim 1 wherein said first processor
30 means is two or more processor means interconnected by a network,

4 The apparatus of claim 3 wherein one of said two or more
processor means interfaces to said user by receiving said
35 query and by presenting said extracted information which is
communicated from the remaining processor means, and wherein
43
the remaining processor means queries said on-line stores and
extracts returned product information.

5 The apparatus of claim 1 wherein said product domain
5 description comprises a list of attributes of products in
said product domain, said attributes being possible product
query parameters,

6 The apparatus of claim 1 wherein said on-line store
10 description for an on-line store comprises:
a. a network address for said product query form of
said on-line store;
b. a mapping of product attributes to fill-in fields
of said product query form;
is C, a first function for recognizing product search
failure pages from said on-line store;
d, one or more second functions for recognizing header
information and trailer information of successful product
search pages from said on-line store; and
e, a third function for recognizing and extracting
product information from said successful product search
pages,

7 An apparatus according to claim 6 wherein said network
25 address, said mapping, said first function, said one or more
second functions, and said third function are represented in
said on-line store description by character strings stored in
said second memory means.

30 8, An apparatus according to claim 7 wherein said character
string for said third function comprises a sequence of HTML
tags and the symbol "text" stored in said second memory
means.

35 9, The apparatus according to claim 1 further comprising a
second processor means for determining said description of an
on-line store by: (i) fetching a plurality of pages from said
44

on-line store; (ii) selecting a plurality of candidate query
forms from said plurality of pages; (iii) for each candidate
query form determining a mapping of product attributes to
fill-in fields of each candidate query form, querying said

5 on-line store with said query form filled-in with dummy products and with popular products to further determine said first function, said one or more second functions, said third function, and estimating ranking of the query success of this query form; (iv) selecting the candidate query form and 10 associated mapping and functions with the highest ranking as the on-line store description; and (v) storing said selected on-line store description in the second memory means.

10 The apparatus according to claim 9 wherein said domain 15 description for the domain of said on-line store comprises a plurality of rules for guiding said determining a mapping of product attributes to fill-in fields.

11 A method for assisting a user in querying for 20 information about a product at one or more on-line electronic stores, said method comprising the steps of:

- a. receiving a user query for said product;
- b. fetching an on-line store descriptions of said on line stores for said product domain;
- C. accessing a product query form at said on-line stores according to said on-line store descriptions;
- d. filling-in said product query forms according to said on-line store descriptions;
- e, submitting said filled-in query forms to said on 30 line stores;
- f. receiving response pages from said on-line stores;
- 9- extracting product information from said response pages according to said on-line store descriptions; and
- h. presenting said extracted information to said user.

12 The method of claim 11 wherein said steps of receiving a user query and presenting said extracted information are 45

performed by a user interface computer, and wherein the remaining steps of said method are performed on one or more computers that are linked among themselves and with the user interface computer by-a network.

13 The method of claim 11 wherein said steps of accessing and submitting are performed in parallel for all said on-line stores.

10 14. The method of claim 11 wherein each of said on-line store descriptions comprises:

- a, a first string representing the network address for said product query form of said on-line store;
- b. a plurality of pairs of strings for mapping of 15 product attributes to fill-in fields of said product query form, one's string of each said pair being the name of a product attribute and the other string of each said pair being the name of a fill-in field of said product query form;
- C. a second string for matching product search failure 20 pages from said on-line store;

d, a third string for matching header information on successful product search pages from said on-line store;
e, a fourth string for matching trailer information of said successful product search pages; and
f, a fifth string for matching and extracting product information from said successful product search pages, said fifth string representing the abstract format of said product information,

30 15, The method of claim 14 further comprising prior to said step of fetching the additional steps of:

a. fetching a plurality of pages from an on-line store without a current on-line store description;

b, selecting a plurality of candidate query forms from
35 said plurality of pages;

C, for each candidate query form determining a mapping of product attributes to fill-in fields of each candidat@e
46

query form, querying said on-line store with said query form filled-in with dummy products and with popular products to further determine said first, second, third, fourth, and fifth strings and said plurality of pairs of strings, and
5 estimating ranking of the query success of this query form; and

d, selecting the candidate query form and associated strings and pairs of strings with the highest ranking as the on-line store description; and

16 The method of claim 15 wherein said abstract format of said product information comprises a plurality of logical lines, and wherein said fifth string matches said logical lines by being an abstract format of said logical lines.

17 The method according to claim 16 wherein said querying with said popular products returns said product search success pages, and wherein said estimating is according to the number of logical lines matched by said fifth string and
20 having extracted text information, price information, or popular product information.

18 The method of claim 15 wherein said querying with said dummy products returns said product search failure pages, and
25 wherein second string matches said product search failure pages,

19 The method of claim 15 further comprising prior to step (a) a step of fetching a product domain description, said
30 product domain description comprising one or more rules for guiding said determining a mapping of product attributes,

20 The method of claim 15 wherein said selecting a plurality of candidate query forms selects one of said
35 plurality of pages as a candidate query form according to

whether said page is near the home page of said on-line

47

store, contains a link likely to lead to a search page, or
whether said page is in the same domain as said home page,

21 A computer readable medium for causing a processor to
5 function according to the method of claim 11.

48

Patent and Priority Information (Country, Number, Date):

Patent: ...19980723

Fulltext Availability:

Detailed Description

Detailed Description

... shopping phase, the vendor

and domain descriptions are used to assist a user in

25 comparison-shopping at an on-line electronic store on the

Internet. In a preferred embodiment in which the vendor

description contains the above described strings, this can be, for
example, purchase

request 655 to find the cheapest price for a product,

Shopping phase modules 651 proceed generally according to the...